

**A207 : 인턴산타**

**포팅 매뉴얼**

삼성청년SW아카데미 서울캠퍼스 7기

자율프로젝트 (6주, 2022/10/11 – 2022/11/18)

강민서(팀장), 김영범, 김혜성, 양경섭, 원재호, 정민호

## [목차]

[1. 기술 스택 2](#_Toc17299)

[2. 빌드 상세내용 4](#_Toc17300)

[3. 배포 특이사항 6](#_Toc17301)

[4. DB 계정 11](#_Toc17302)

[5. 프로퍼티 정의 12](#_Toc17303)

[6. S3 server 13](#_Toc17304)

# 프로젝트 기술 스택

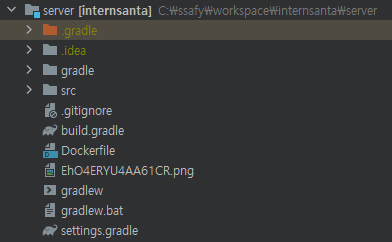
1. 이슈관리 : Jira
2. 형상관리 : GitLab
3. 커뮤니케이션 : MatterMost, Notion
4. 개발 환경
   1. OS : Windows 10
   2. IDE
      1. IntelliJ IDEA 2021.2.4
      2. Visual Studio Code 1.70.1
   3. Database : MySQL WorkBench 8.0 CE
   4. Server : AWS EC2 (MobaXterm)
      1. Ubuntu 20.04.4 LTS
      2. NginX 1.18.0
      3. Docker 20.10.17
5. 상세 사용
   1. Back-end
      1. Java (Open JDK zulu8.33.0.1)
      2. Spring Boot 2.5.6
      3. Gradle 7.4.2
      4. Lombok 1.18.20, Swagger 3.0.0, JPA Hibernate
   2. Front-end
      1. HTML5, CSS3, JavaScript
      2. React 16.8.0, Recoil 0.7.4,

**C.** Node.js 16.14.2

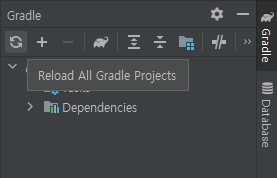
**D.** Styled Component 5.3.5, Three.js, React Three Fiber 8.8.10

* 1. 추천 알고리즘, openCV
     1. Python 3.9.12
     2. Django 4.1.1
     3. Anaconda 4.12.0
  2. AWS S3 2.3.1

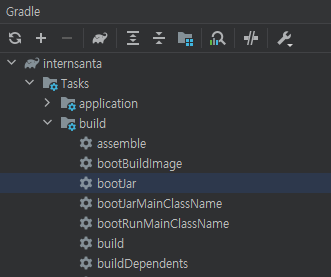
# 빌드 상세내용



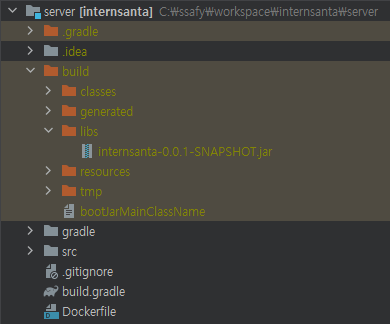
IntelliJ에서 file > open > server 폴더 선택합니다. Gradle Daemon 작업이 자동으로 수행되고, 끝나면 우측의 Gradle 탭을 열어 ‘Reload All Gradle Projects’를 클릭하여 실행합니다.



1. Internsanta Backend 빌드
   1. Gradle -> Internsanta -> Tasks -> build 순으로 이동하여 BootJar를 더블클릭합니다.



* 1. 다음과 같이 Jar 파일이 정상적으로 만들어진 것을 확인할 수 있습니다.



1. Internsanta Frontend 빌드
   * 1. node\_modules를 위한 기본 install

npm install

* + 1. 빌드하기

npm run dev

# 배포 특이사항

1. MobaXterm으로 ssh 세션 접속
2. Docker 설치블로그<https://shanepark.tistory.com/237>참고
3. NginX 설치 & certbot으로 발급받은 인증서 세팅
4. NginX 설치

$ sudo apt install nginx

1. certbot 이용해 SSL 인증서 발급

$ apt-get install python3-certbot-nginx

$ certbot certonly --nginx -d <도메인 주소>

1. NginX 설치된 경로로 이동해서 설정 파일 수정

$ sudo vim nginx.conf

|  |
| --- |
| # /etc/nginx/sites-available/internsanta.conf  server {  listen 80; #80포트로 받을 때  server\_name <도메인 주소>; # 없을경우 localhost  return 301 https://<도메인 주소>$request\_uri;  }  server {  listen 443 ssl http2;  server\_name <도메인 주소>;  ssl\_certificate /etc/letsencrypt/live/<도메인 주소>/fullchain.pem; ssl\_certificate\_key /etc/letsencrypt/live/<도메인 주소>/privkey.pem;  location / {  proxy\_pass http://localhost:3000;  }    location /api/v1 {  proxy\_pass http:// <도메인 주소>:8081;  proxy\_redirect off;  charset utf-8;    proxy\_http\_version 1.1;  proxy\_set\_header Connection "upgrade"; proxy\_set\_header Upgrade $http\_upgrade; proxy\_set\_header Host $http\_host;  proxy\_set\_header X-Real-IP $remote\_addr;  proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for; proxy\_set\_header X-Forwarded-Proto $scheme;  proxy\_set\_header X-NginX-Proxy true;  } |
| location /api/v2 {  proxy\_pass http://localhost:8000;  proxy\_redirect off;  charset utf-8;  }  } |

1. Docker에 MySQL 설치

$ docker pull mysql:8.0.29

$ docker run -d -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=<비밀번호> -name internsanta-mysql mysql:8.0.29 --character-set-server=utf8mb4 –

collation-server=utf8mb4\_unicode\_ci

1. Frontend 도커 이미지 생성 및 실행

(1) internsanta Frontend 폴더에 Dockerfile 생성

# nginx 이미지를 사용합니다. 뒤에 tag가 없으면 latest 를 사용.

FROM nginx

# root 에 app 폴더를 생성

RUN mkdir /app

# work dir 고정

WORKDIR /app

# work dir 에 dist 폴더 생성 /app/dist

RUN mkdir ./ dist

# host pc의 현재경로의 dist 폴더를 workdir 의 dist 폴더로 복사

ADD ./ dist ./ dist

# nginx 의 default.conf 를 삭제

RUN rm /etc/nginx/conf.d/default.conf

# host pc 의 nginx.conf 를 아래 경로에 복사

COPY ./nginx.conf /etc/nginx/conf.d

# 80 포트 오픈

EXPOSE 80

# container 실행 시 자동으로 실행할 command. nginx 시작함

CMD ["nginx", "-g", "daemon off;"]

(2) npm install 후 도커 이미지 생성 후 실행

$ npm run dev

$ docker build -t nginx-react:0.1 .

$ docker run --name internsanta-frontend -d -p 3000:80 internsanta-frontend:0.1

6) Backend 도커 이미지 생성 후 실행

1. internsanta Backend 폴더에 Dockerfile 생성

FROM java:8

VOLUME /tmp

EXPOSE 8081

ARG JAR\_FILE=internsanta-0.0.1-SNAPSHOT.jar

COPY ${JAR\_FILE} app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

ENV TZ=Asia/Seoul

RUN apt-get install -y tzdata

1. gradle build하고 도커 이미지 생성 후 실행

$ gradle clean build

$ docker build -t internsanta-backend:0.0.1 .

$ docker run --name internsanta-backend -d -p 8081:8081 internsanta -backend:0.0.1

7) Django 도커 이미지 생성 후 실행

1. internsanta Algo 폴더에 Dockerfile 생성

FROM python:3.9.12

RUN pip3 install django

WORKDIR /usr/src/app

COPY . .

# WORKDIR ./internsanta

RUN pip install --upgrade pip # pip 업그레이드

RUN pip install -r requirements.txt # 필수 패키지 설치

EXPOSE 8000

CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

1. 도커 이미지 생성 후 실행

$ docker build -t internsanta-algo:0.0.1 .

$ docker run --name internsanta- algo -d -p 8000:8000 internsanta - algo:0.0.1

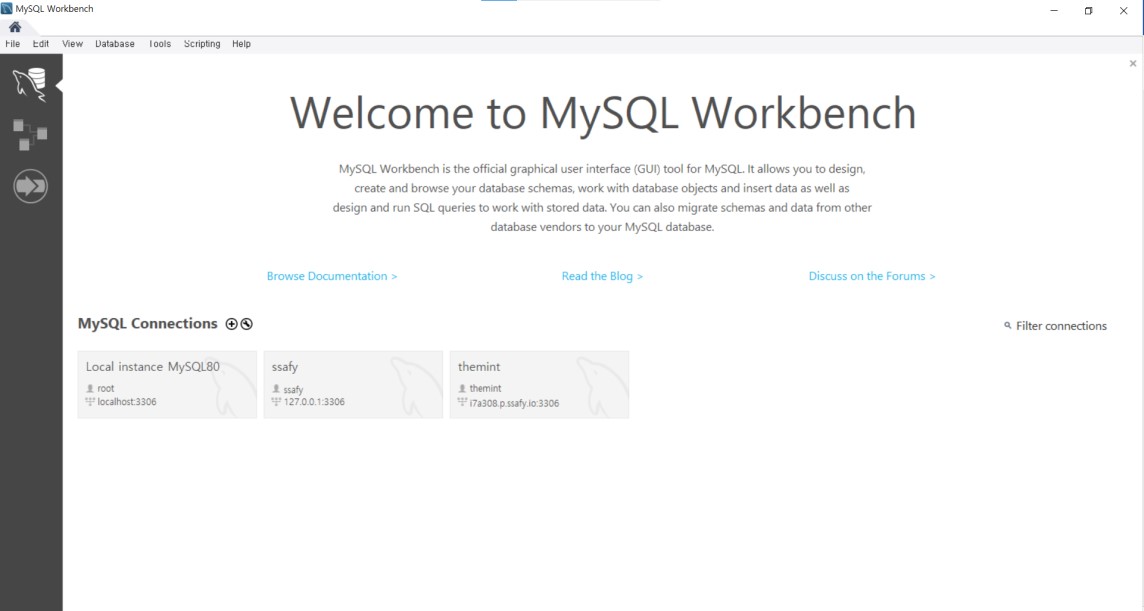
\* 위의 모든 과정은 Jenkins를 통한 자동 배포로 대체 완료

# DB 계정

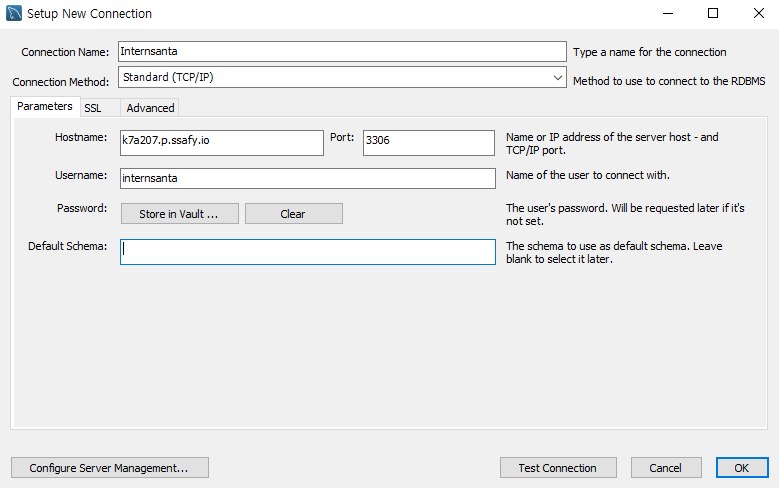
1)

MySQL WorkBench

추가하기



2) EC2 지정 정보 넣기



2. 도메인과 포트 번호 입력

5. 생성

4. 클릭 후 비밀번호 입력

3. 계정 명 입력

1. Connect 이름 설정

- username : internsanta, password : internsanta207

# 프로퍼티 정의

1. Spring

# jwt

jwt.secret={secretkey}

jwt.expiration=10800000

1. Django

SECRET\_KEY = my.settings.SECRET\_KEY

# S3

AWS\_REGION = my\_settings.AWS\_REGION

AWS\_STORAGE\_BUCKET\_NAME = my\_settings.AWS\_STORAGE\_BUCKET\_NAME

AWS\_ACCESS\_KEY\_ID = my\_settings.AWS\_ACCESS\_KEY\_ID

AWS\_SECRET\_ACCESS\_KEY = my\_settings.AWS\_SECRET\_ACCESS\_KEY

# file upload size

MAX\_UPLOAD\_SIZE = 5242880

DATA\_UPLOAD\_MAX\_MEMORY\_SIZE = None

FILE\_UPLOAD\_MAX\_MEMORY\_SIZE = 5242880

DATA\_UPLOAD\_MAX\_NUMBER\_FIELDS = None

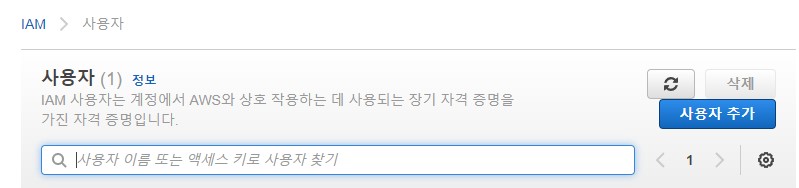
# S3 server

1) IAM 계정 생성

(1)

사용자

추가



(2)

사용자

이름

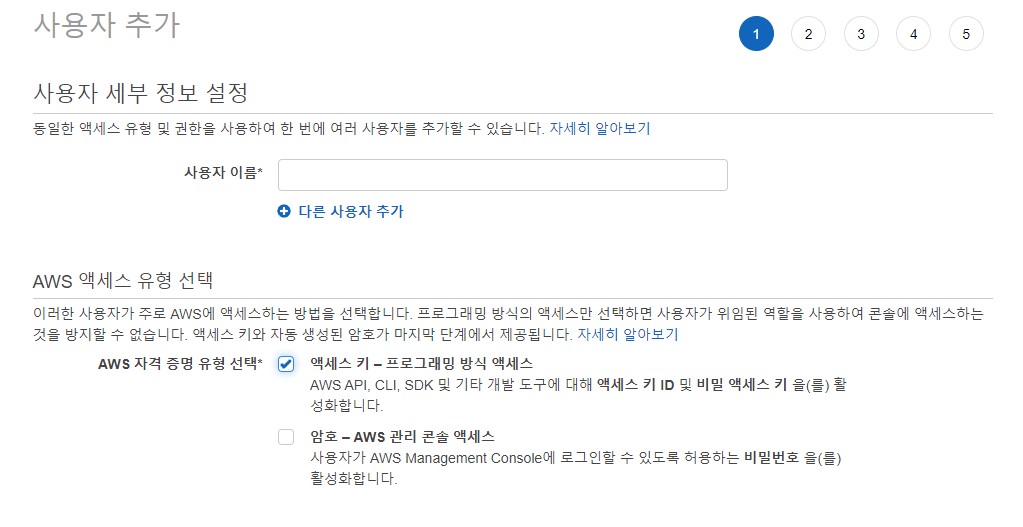
입력

,

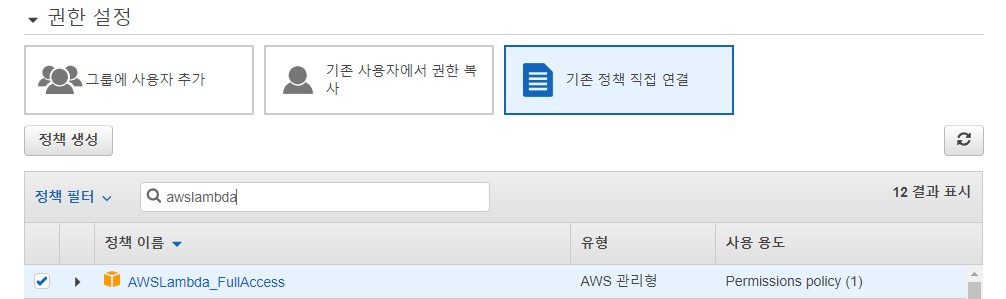
엑세스

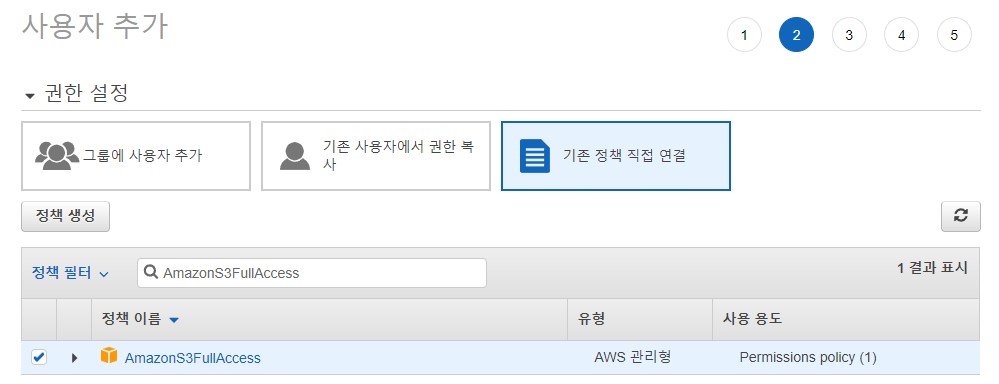
키

체크



1. AWSLambda\_FullAccess, AmazonS3FullAccess 정책





1. 액세스 키 ID, 비밀 엑세스 키 저장

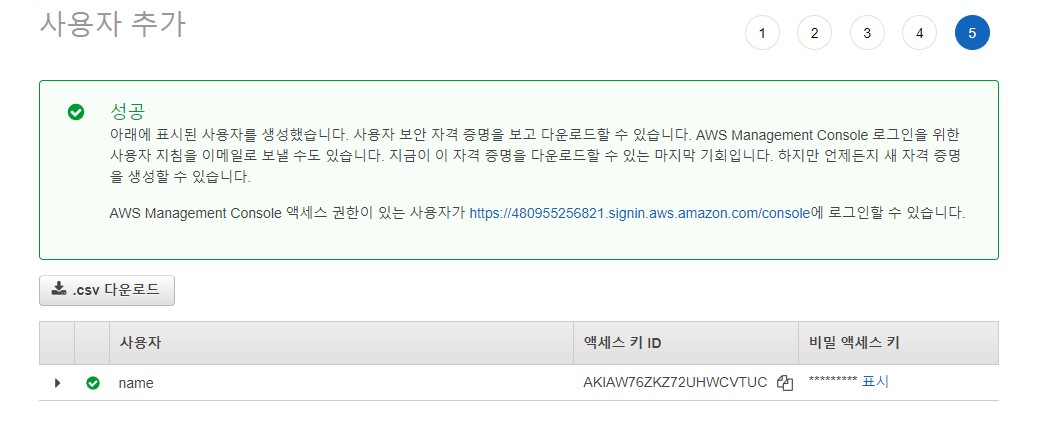
2)

역할

(

Role)

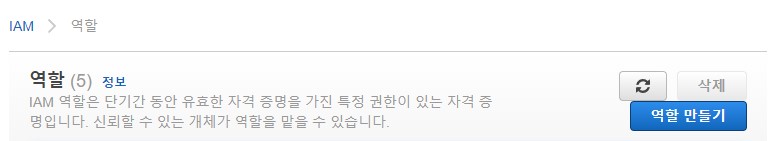
설정



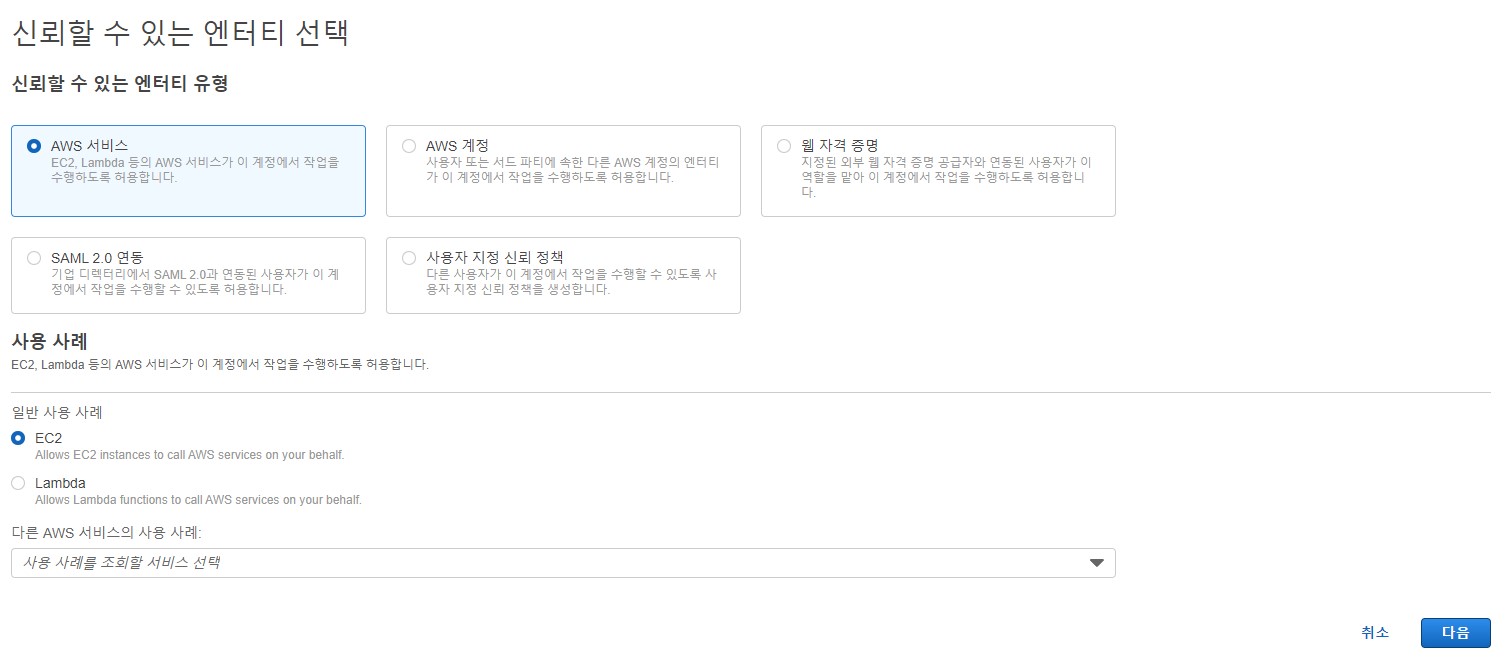
(1)

역할

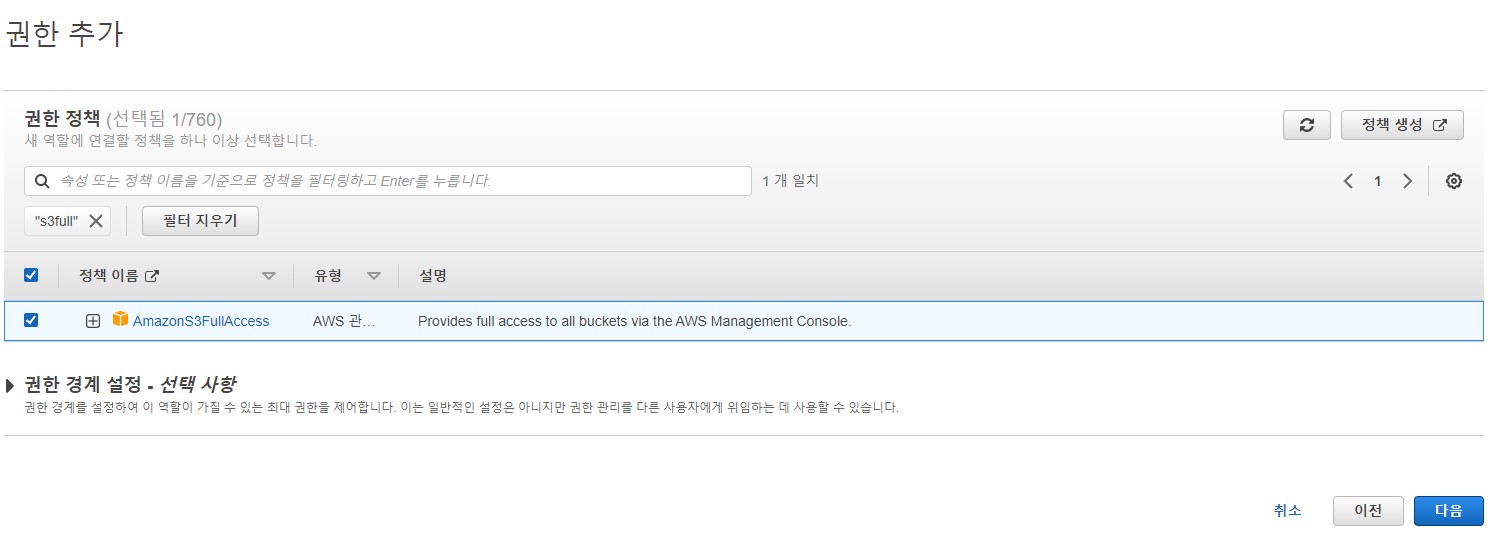
만들기



1. 엔티티 선택



1. 권한 추가



(4)

역할

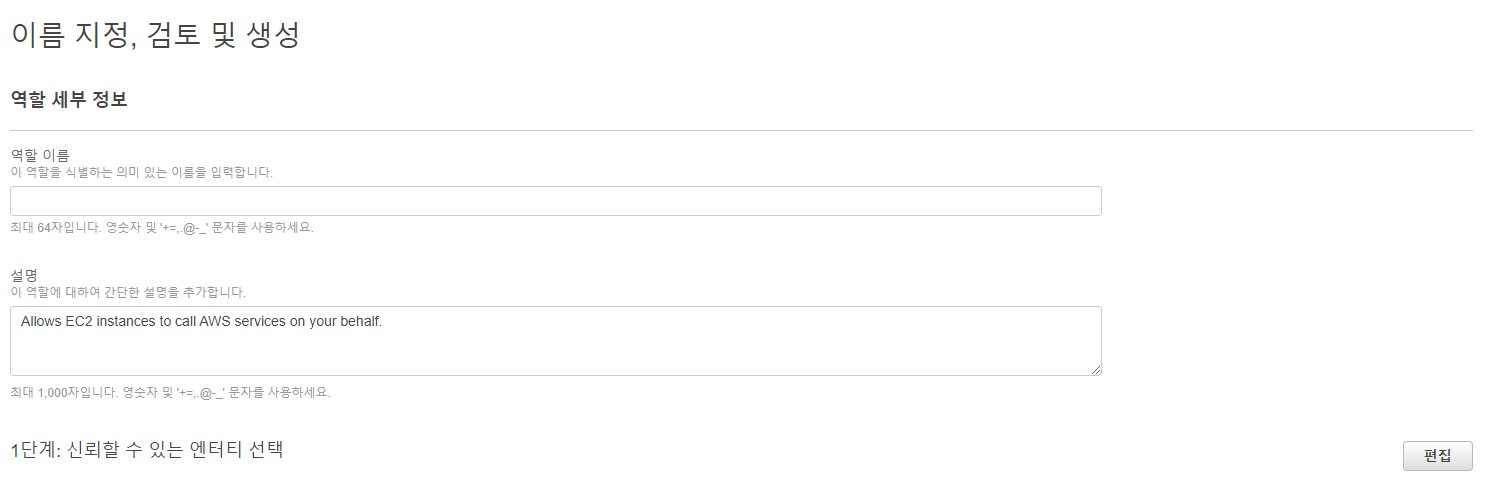
이름

설정

후

역할

생성



1. Cognito

(1)

C

ognito

자격

증명

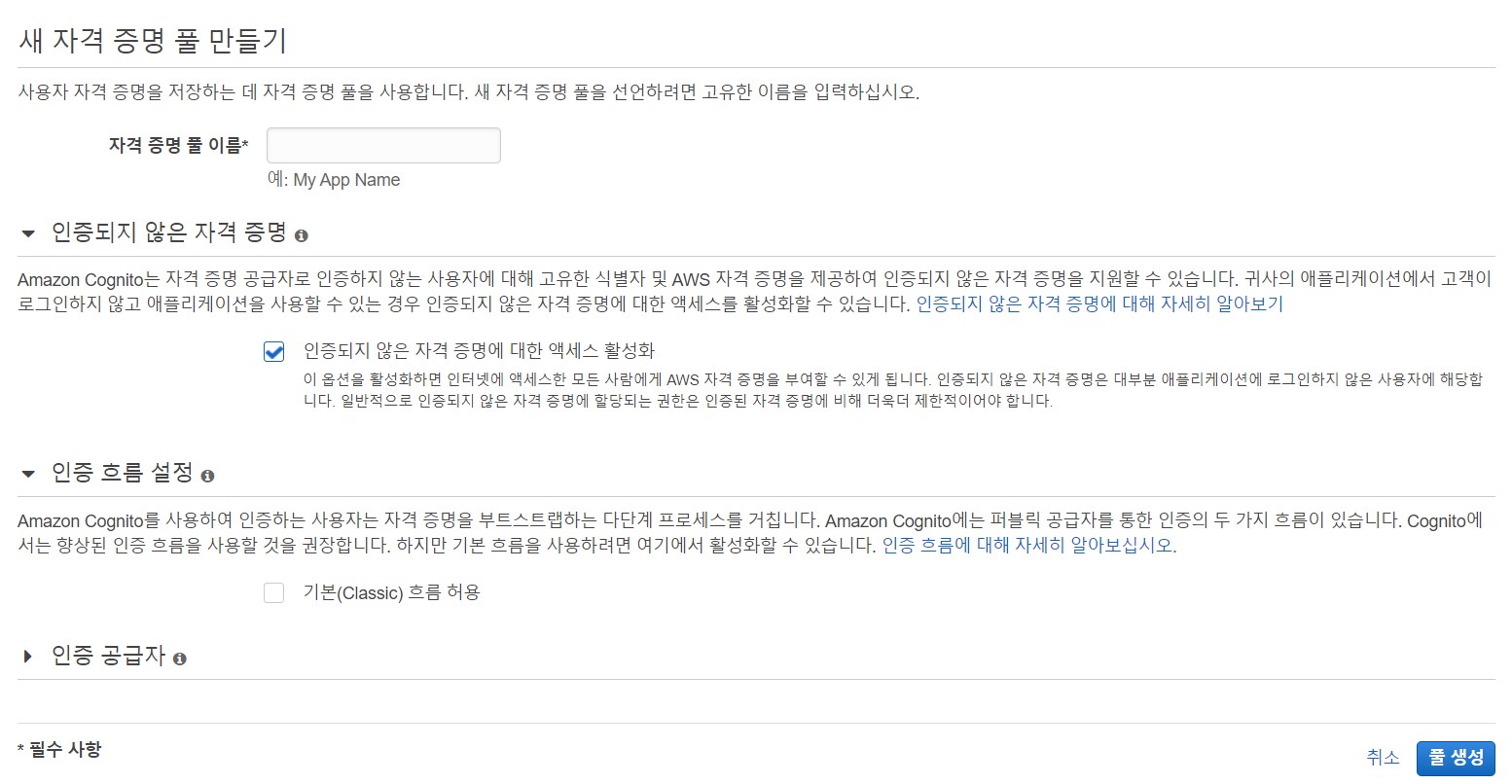
풀

관리

클릭



* 1. 자격 증명 풀 이름 입력, 인증되지 않은 자격 증명에 대한 엑세스 활성화 체크 후 풀 생 성



1. S3 버킷 생성

(1)

버킷

만들기

(2)

버킷

이름

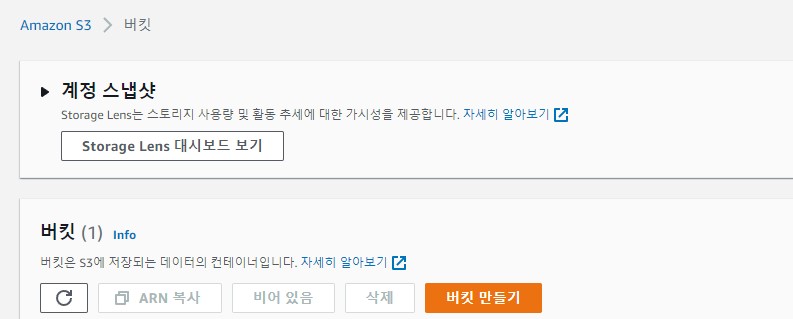
입력

,

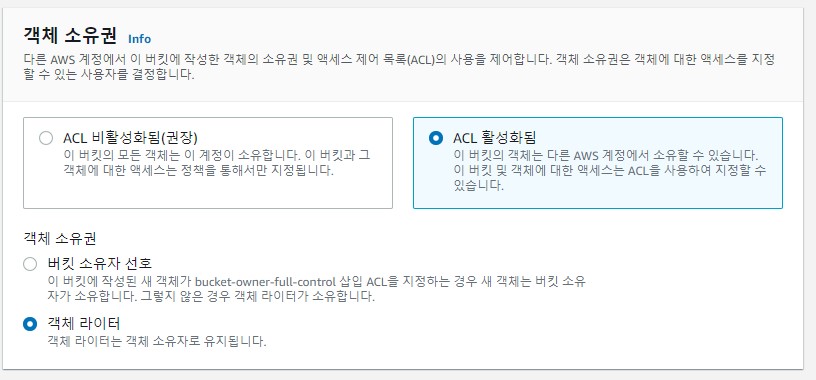
리전

서울로

설정



* 1. ACL 설정



* 1. 퍼블릭 엑세스 차단 설정 후 버킷 생성



* 1. 버킷 정책 설정



* 1. CORS 설정

